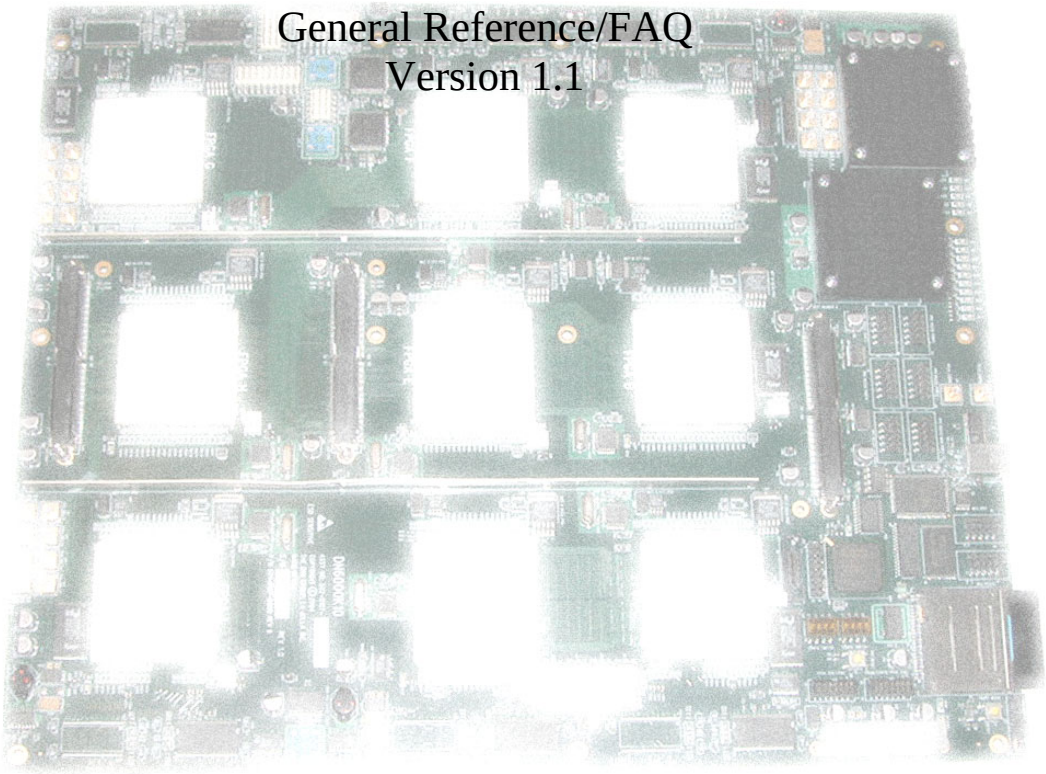
DINI Group DN6000K10 Series



General Reference/FAQ
Version 1.1

Table of Contents

Checklist for Troubleshooting/Debugging FPGA Designs 3
General FAQ 7
DCM-Related Information 8
Roboclock Information 9

Checklist for Troubleshooting/Debugging FPGA Designs

1. Read through all of the synthesis and place & route warnings. Look for removed logic that shouldn't have been removed and warnings about ignored timing constraints.
2. Check to make sure all pins are constrained to a fixed location. This means, in the project's .par file, look for a line that says

Number of LOCed External IOBs 702 out of 702 100%

If the percentage listed is not 100%, this means some signals were not constrained in the UCF file. If this is the case, look at the .pad file to find which pins could not be located. In this case, it helps to import it into Microsoft Excel. To import the file into Microsoft Excel,

a. Open the file in Microsoft Excel. Excel will bring up a dialog box, asking what kind of file it is opening. Select *Delimited,* and hit next. On the next screen, you will see an option to select the delimiter. Make sure *Other* is checked, and put "|" as the delimited. Finish the dialog, and the file should be imported into Excel.
b. Scroll down, and select any cell past the ----- line.
c. Go to the *Data* menu, and select *Sort*. For the first *Sort By* field, select Column M, the column corresponding to *Constraint*. Then, select *Signal Name.*

All of the pins used by the design should be marked LOCATED in column M. To see pins that were not constrained, scroll to the bottom of the LOCATED pins. You will see one or more pins with signal names, but without the LOCATED tag in column M. These are the pins that are not constrained.

Some IOB pins will be unused intentionally. To avoid problems with these pins, in the Xilinx Project Navigator, right-click on "Generate Programming File" and select "Properties," then click on the "Configuration Options" tab. Near the bottom there is an option for "Unused IOB Pins." That option should be set to "Float." As a general rule, that option should ALWAYS be set to "Float." Don't worry about pins that are not driven, they will stabilize either high or low and stay there.

3. Make sure all clocks have a period constraint. In the .par file, you will see a table of constrained clocks.

```
--------------------------------------------------------------------------------
Constraint                                | Requested  | Actual     | Logic
                                          |            |            | Levels
--------------------------------------------------------------------------------
NET "i_pci_clk_ibuf/IBUFG" PERIOD =  30 n | 30.000ns   | 20.139ns   | 0
S   LOW 50.000000 %                       |            |            |
--------------------------------------------------------------------------------
TS_cclk = PERIOD TIMEGRP "cclk"  15 nS    | 15.000ns   | 14.290ns   | 4
HIGH 50.000000 %                          |            |            |
--------------------------------------------------------------------------------
TS_dclk = PERIOD TIMEGRP "dclk" TS_cclk * | N/A        | N/A        | N/A
```

0.500000 HIGH 50.000 % | | |

Verify that all of the clocks utilized by your project are constrained.

4. Make sure all outputs have a *clk to pad* constraint. For this, look in the .twr file. You should see several tables, titled *Clock *clk to Pad*, where * stands for the letter corresponding to your clock. This number is the amount of time it takes between the time a part receives a clock signal, and the time that the output corresponding to the clock input is updated. Verify that all of your outputs have valid *clk edge to PAD* numbers.

5. Verify that all inputs have setup/hold contraints. For this, look again into the .twr file. Observe the tables titled *Setup/Hold to clock*. Verify that *setup to clock edge* and *hold to clock edge* numbers are valid.
   These numbers describe when a particular input signal will be assigned to a register, with the *setup to clock edge* corresponding to the time *before* the clock edge the signal must be at the input, and the *hold to clock edge* corresponding to the time *after* the clock edge must be held at the input. Usually, this will correspond to a rising clock edge, indicated by an *(R)* next to the time. For DDR parts, there will also be an *(F)* time, corresponding to the falling clock edge.

6. Make sure that any constraints actually constrain a valid trace. To verify, you should check inside the .twr file. If a timing constraint says *0 items analyzed*, you should check that it does constrain something. Please note, some constraints may say *0 items analyzed* even if they are valid; an example is a constraint placed on a pad before a DCM.

7. Verify that the FPGAs have been programmed. Do this by checking with a computer plugged into the board's MCU RS232 port:

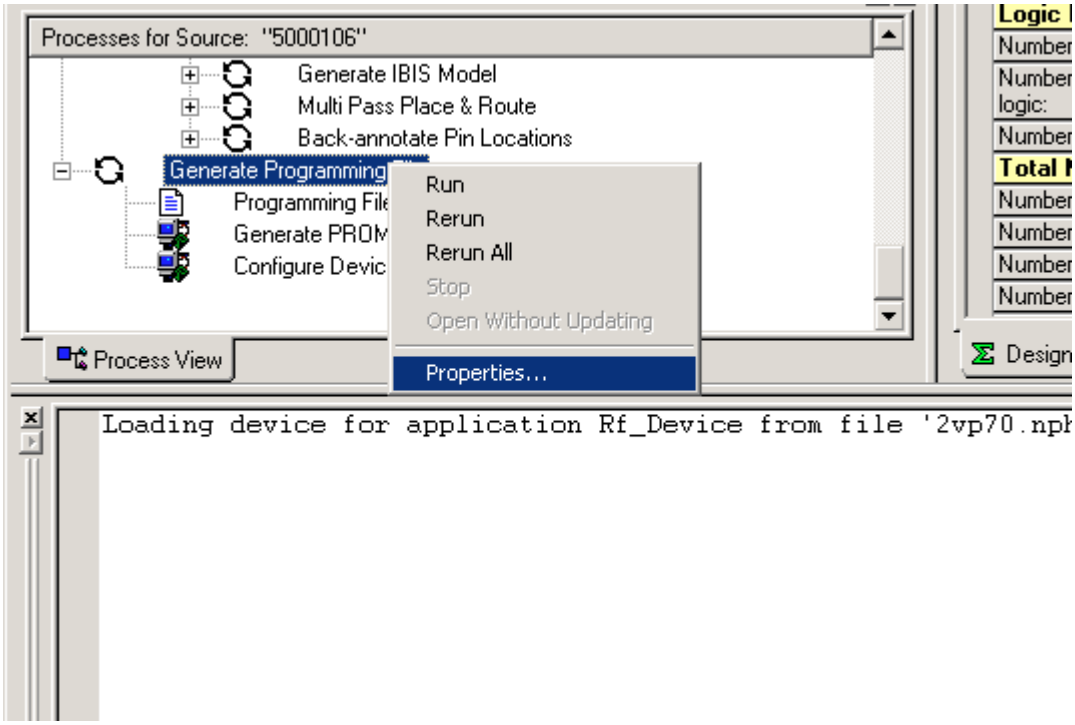
   1. Set up the board as described in your user manual, with the computer's COM port connected to the MCU 232 port on the board.
   2. Open Hyperterminal on the computer, make sure it is listening to the right port, and turn the board on. Again, look in the user manual for more detailed instructions.
   3. The board will output to the COM port what bit files were loaded on power up. Make sure the right bitfiles are loaded, and no errors are presented.

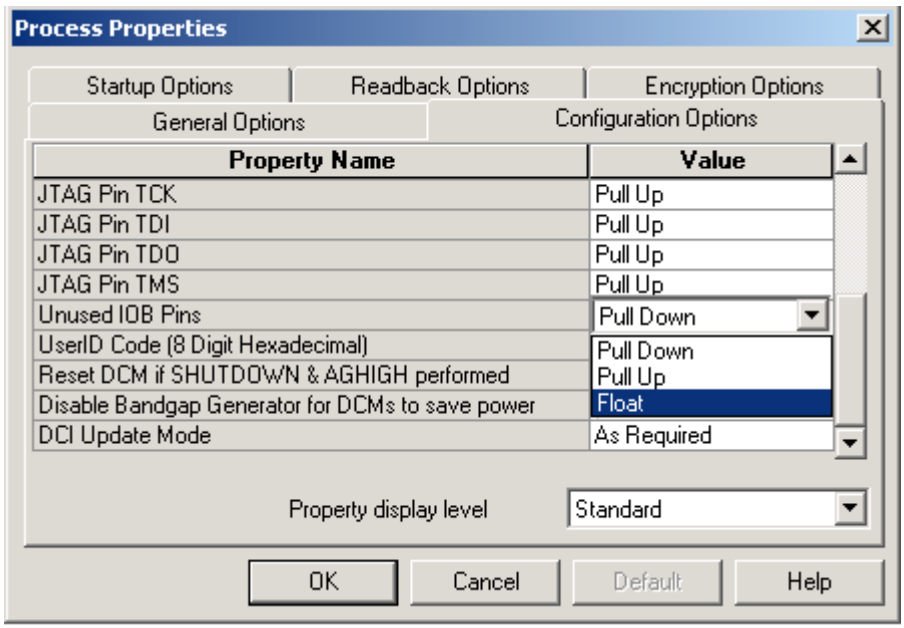
   For details, see Chapter 6 Part 3 ("Configuring Hyperterminal") in your board manual.

8. Make sure that the input/output flipflops have been implemented, when necessary. For this, see your .mrp file. Section 6 (*IOB Properties*) in the .mrp file has a table which contains the registers each signal is assigned to. All bi-directional signals should have one input flip-flop (INFF), one output flipflop (OFF), and an enable flipflop (ENFF). Input signals will only have an input (INFF) flip-flop. Output signals will have an output flip-flop (OFF), and possibly an enable flip-flop

(ENFF).

9. Verify that all IO pins are assigned the correct IO standard. To check this, look at the Excel spreadsheet you have generated for item #2 of this list. There is a column called *IO Standard*, this is the IO standard specified for each pin. Verify that it is the correct IO standard.
   The proper standard will depend on the board that you are using, and your application. For a connection between two FPGAs, the only requirement is that the IO standard is the same on both ends. For a connection between an FPGA and another component, the IO standard required is particular to that component. For example, a connection between an FPGA and a PCI slot would have the standard *PCI66_3*, whereas for a DDR, it could be either *SSTL1* or *SSTL2*, depending on the signal type.
10. Verify that your Unused IOBs are set to float. In the ISE, right-click on “Generate Programming File.” Select properties, and under the Configuration tab, set Unused IOBs to “Float”.





If they are set to pull-down, then unassigned pins on an FPGA will be driven low. This is particularly bad on boards with multiple FPGAs, as main bus pins could be driven low, preventing communication between FPGAs and other devices.

11. Check to see that the clock(s) you use are running. An easy test for this is to assign the incoming clock to an output pad on the FPGA. Then, probe the daughtercard pin connected to the output pad with an oscilloscope to verify that the clock is being sent to the FPGA.

For Troubleshooting Daughtercard-Related Problems

- Double-check that the connections match between your FPGA pins and the daughtercard pins. Reference the Daughtercard Connection Summary on our website *(http://www.dinigroup.com/product/data/obsdaughter/files/conn_summary.xls)* for details on the daughtercard header pinouts.
- Make sure that none of the other FPGAs are driving daughtercard-connected MB pins. Check for logic in your source code, and make sure that the "Unused IOBs" option in the ISE settings is set to "Float." If it is set to "Pulldown," then those FPGAs are driving any pin that is not assigned in the source code. *See item #10 above.*
- If the connections are on J3 and/or J4 on the daughtercard, make sure the OE pins on the daughtercard buffers are active. The OE pins must be driven low to allow for communication via the buffered connections. Refer to the daughtercard schematic for details.

General FAQ

Q. I'm having a tough time pushing data quickly between any two FPGAs on the DN3000K10 board. I can't seem to push it beyond 66 MHz. Do you have (fragments of) an application that illustrates how you handle these transfers? Specifically, the properties of your drivers, clocking strategies, where things are registered, etc.

A. The memory-test reference design on the source CD has samples of FPGA transfers that we have run successfully at 100 MHz. Have you checked the timing reports of your designs? They will give clock-to-pad and setup times for outputs and inputs. We usually assume 1 ns for board transit and 1 ns for jitter, so you should be able to estimate the maximum frequency from there. The most crucial thing for high-speed IO is registers packed in the IOB. Check the map report (.mrp) to make sure that you have output registers on all outputs, OE registers for those that can be tri-stated, and input registers on all inputs. If not, you need to make sure that no logic exists between the pads and their registers. You may also need to "preserve" output and OE registers, because some synthesis tools will optimize registers out if there is another register with identical inputs, which creates a single register that can't be packed in either IOB.
Don't forget that you should be using a DCM to deskew the internal clock tree in the FPGA. This will allow you to reduce the latency between the pad and the register.

Q. I am unable to connect to the PowerPC on my board in a stand-alone configuration. I would like to run the RocketIO tests.

A. This problem is most often caused by an improperly configured clock. If the PowerPC does not have the correct clock input (as specified in the projects settings), it won't run at all. Since it never reads or writes to the UART module, no data comes through. The RocketIO design requires setting CCLK (one of the Roboclock output banks) to 66 MHz. A Xilinx DCM is used to multiply by 3, to provide a 200 MHz clock to the PowerPC. If it doesn't get the 200 MHz input, it will freeze.

Q. I recently formatted my SmartMedia card under Windows, and now the board is unable to access it.

A. Windows cannot correctly format SmartMedia cards. However, there is a utility that can do this correctly. Download the utility at
http://www.dinigroup.com/product/common/SMFormat.zip.
Using SMFormat, you can format SmartMedia cards so that they can be read by the board.

DCM-Related Information

DN6000K10PCIe – Known Schematic Error

The schematic has the DCM at X0Y1 labeled as "X0Y0" even though the one at X0Y0 is labeled correctly. The schematic has no effect on DCMs; they are only included as a reminder. They are also not shown in position or to scale, so if you are working on DCM placement you should use the Virtex-II Pro Datasheet as a reference, or looking at a previous design (.ncd file) in FPGA Editor. Either of those will show the true placement of the DCMs, instead of a distorted graphical representation.

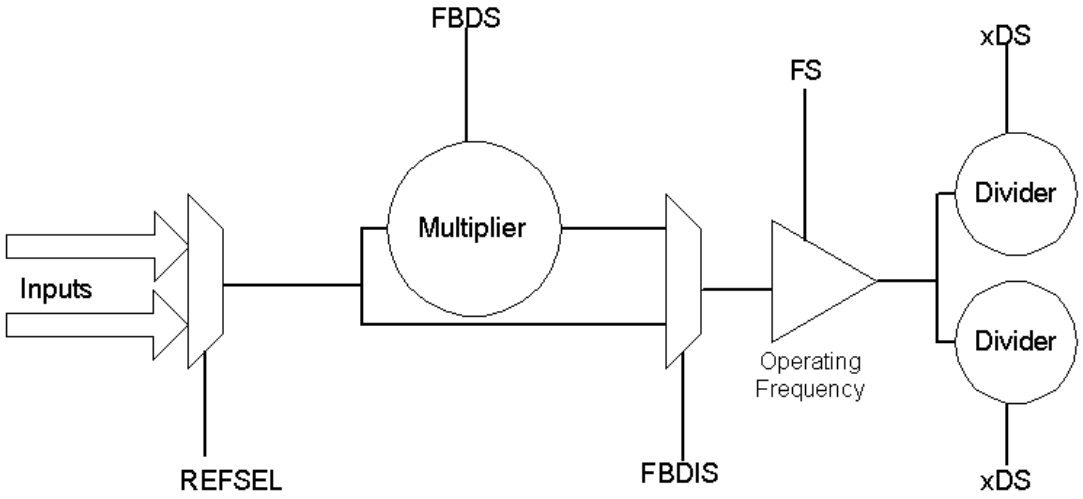
Virtex-II family DCM Phase Control Information

The Xilinx DCM has a "coarse" phase control feature for 90-degree phase shifts (using CLK0, CLK90, etc), and a "fine phase shift" feature that can shift all outputs by smaller amounts. The phase counter, internal to the DCM, can adjust in increments of 1/256 of the clock period. Our dcm_phase_shift module has a counter that tracks the internal phase counter, and compares it to a register that can be written by the PCI/USB/PowerPC interfaces. So, if you write the phase you want, our state machine will update the DCM one step at a time until the registers match.

The "input_clk" port into the dcm_phase_shift module is the clock for register writes/reads, so it has to match the clock for the data interface to PCI/USB/PowerPC. The "ps_clk" port clocks all the DCM phase shift control signals. Some problems can occur if ps_clk comes from a DCM output, so it is best to connect it to a non-DCM clock.

Roboclock Information

The Roboclock functions as a selectable input with multiply and divide capabilities. A set of jumpers controls the function of a Roboclock. The following is an explanation of jumpers and their effect on Roboclock function.



Roboclock internal multiply/divide setup (for one input)
Note: This is not an exact representation of the function of a Roboclock. It is intended for reference in this document only.

Jumper Functionality:

FBDS0 / FBDS1	Adjusts the multiplier on the Roboclock. This multiplier is global to all banks on the Roboclock. See *Table 4* of the Roboclock Datasheet for jumper setups.
xDS0/xDS1 (*i.e. DDS0/DDS1)*	Adjusts the divider on the Roboclock. Note that this divider is specific to a particular bank on the Roboclock. See *Table 4* of the Roboclock Datasheet for jumper setups.
REFSEL	Selects the Roboclock input. See next page for more information.
FBDIS	Disables the feedback output. Effectively sets the multiplier to 1.
FS	Selects the frequency range for the operating frequency. Note that this is the frequency after the multiplier, but before the dividers. Also, the maximum output after the dividers is 185Mhz. See *Table 1* of the Roboclock Datasheet for jumper setups.
MODE	Determines what the Roboclock does with an illegal input. These jumpers should not be changed. Note that setting it to medium will affect Roboclock function even when the Roboclock is provided a legal input.

In the stock configuration on DN6000K10-series boards, the Roboclocks will be taking the 33Mhz oscillator signal, and multiplying it by 4 to get a net clock frequency of 133Mhz.

The REFSEL can have one of two positions. When driven low, it will use the REFA pair. When driven high, it uses the REFB pair. Each pair can function either as a differential pair, or as two single-ended inputs. Therefore, two single-pin clocks can be hooked up to each reference pair, as long as only one is active.

One notable example is the REFSEL functionality for REFA on Roboclock 2. If the REFA pair is selected, you have a choice of two inputs, controlled by a separate jumper on the board (J2 on the DN6000K10Se). With the jumper enabled, the REFA pair on Roboclock will be receiving an output from Roboclock 1 (DCLK3 on the DN6000K10Se). Otherwise, it receives a signal directly from the FPGA (FPGA_GCLKOUT on the DN6000K10Se). Please see Chapter 2 Section 3 of your product manual for details specific to your board.

Table 1. Frequency Range Select

FS[2]	CY7B993V f_{NOM} (MHz)		CY7B994V f_{NOM} (MHz)	
	Min.	Max.	Min.	Max.
LOW	12	26	24	52
MID	24	52	48	100
HIGH	48	100	96	200

Table 1 of the Roboclock Datasheet – Operating Frequency selection via FS

Table 4. Output Divider Function

Function Selects		Output Divider Function				
[1:4]DS1 and FBDS1	[1:4]DS0 and FBDS0	Bank 1	Bank 2	Bank 3	Bank 4	Feed-back Bank
LOW	LOW	/1	/1	/1	/1	/1
LOW	MID	/2	/2	/2	/2	/2
LOW	HIGH	/3	/3	/3	/3	/3
MID	LOW	/4	/4	/4	/4	/4
MID	MID	/5	/5	/5	/5	/5
MID	HIGH	/6	/6	/6	/6	/6
HIGH	LOW	/8	/8	/8	/8	/8
HIGH	MID	/10	/10	/10	/10	/10
HIGH	HIGH	/12	/12	/12	/12	/12

Table 4 of the Roboclock Datasheet – Multiplier and Divider Setup